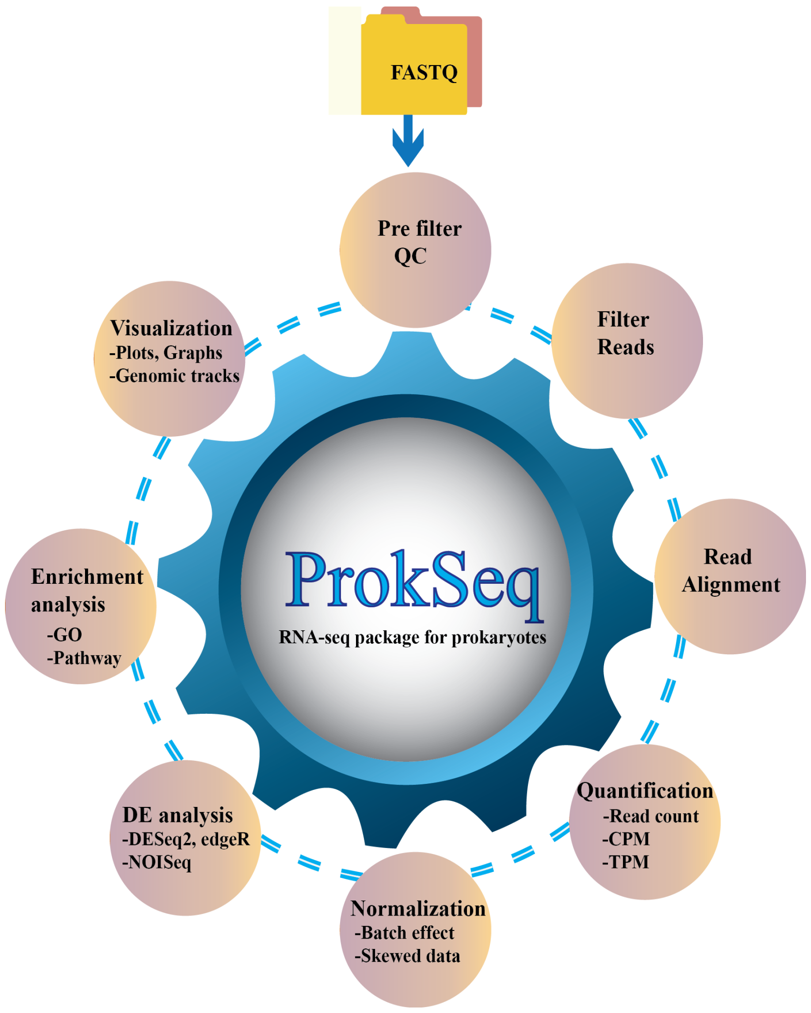
prokSeq

Release 2.0

Apr 05, 2022

Contents:

ProkSeq is an automated RNA-seq data analysis package for Prokaryotic, where users can perform all the necessary steps of RNA-seq data analysis from quality control to pathway enrichment analysis. It has a wide variety of options for differential expression, normalized expression, visualization, and quality control, and publication-quality figures. It is also less time consuming as the user does not need to observe and control the analysis process. The user needs to specify the descriptions of the samples and define the parameter file accordingly. ProkSeq also automatically do the quality filtering of the bad reads and run the analysis on good quality reads.



CHAPTER 1

DOWNLOAD:

The pipeline can be obtained from the following repositories. [GitHub], [Docker], and [Anaconda Cloud].

CHAPTER 2

DOCKER:

We strongly recommend using docker to run the pipeline. The external dependencies and R dependencies are all bundled in the container. The container prokseq-v2.0:latest is available in https://hub.docker.com/repository/docker/snandids/prokseq-v2.0

Step 1: To pull the image from the Docker Hub registry:

```
docker pull snandids/prokseq-v2.0:latest
```

Step 2: To Run

```
docker run -it snandids/prokseq-v2.0:latest
sh-5.0# cd prokseq
```

Step 3: Activate the environment

```
sh-5.0# source /etc/profile.d/conda.sh
sh-5.0# conda activate py36
(py36) sh-5.0# <YOU WILL GET THIS PROMPT>
```

Step 4: Run the example Run the pipeline with the example files

```
(py36) sh-5.0# python scripts/prokseq.py -s samples.bowtie.PEsample -p param.bowtie.
↪yaml -n 4
```

The script will run with PE (paired-end) samples described in samples.bowtie.PEsample, and with the parameters defined in param.input.yaml. The program is submitted with four processors.

Step 5: How to manage data within your Docker containers.

Once the ProkSeq pipeline is working successfully with the example files, one can go for the real data.
There are two ways to manage data within your Docker containers.

- **A. Method 1** - Using volume mounts <br/>

- **B. Method 2** - Docker cp

A. Method 1:

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.
Exit from the existing docker and mount the container with a volume.

```
docker run -it -v /home/user/prokseqData:/root/prokseq/ --name PROKSEQ snandids/
↪prokseq-v2.1:v1
```

Assuming the user has all the required data files (sample[fq/fastq], GTF, BED, genome/transcript files, etc) in the folder */home/user/prokseqData*, the above command will mount the directory */home/user/prokseqData* in */root/prokseq/* inside container.
User can run the pipeline with appropriate sample files and parameter files inside the docker.

```
sh-5.0# cd prokseq
sh-5.0# source /etc/profile.d/conda.sh
sh-5.0# conda activate py36
(py36) sh-5.0# <YOU WILL GET THIS PROMPT>
```

Modify the parameter file with appropriate path and desired parameters.
Modify the sample file with the fastq file names and the genome/transcript file names.

```
(py36) sh-5.0# python scripts/prokseq.py -s samples_def_file -p param_def_file -n 4
```

Once done, the user can exit the docker.

```
(py36) sh-5.0# exit
```

The output will remain in the */home/user/prokseqData*.

To remove the container.:

```
> docker container rm PROKSEQ
```

B. Method 2:

To copy the files (sample[fq/fastq], GTF, BED, fasta (Genome/transcript), etc) to and from the container:
Find out the containerID from another terminal. For example:
Run the command

```
docker ps -a
```

Output (somewhat similar)

```
CONTAINER ID        IMAGE                          COMMAND             CREATED             ␣
↪ STATUS                  PORTS                   NAMES
8f780c0a9969        snandids/prokseq-v2.1:v1      "sh"                5 minutes ago       ␣
↪ Up 5 minutes                                    fervent_feynman
```

Then follow the following:
Copy to the container:
In the example below, local test.fasta file is copied to the container *(/root/prokseq/)*.

```
docker cp test.fasta 8f780c0a9969:/root/prokseq/test.fasta
```

Copy from the container:
In the example below, a file from */root/prokseq/* will be copied to the local working directory.
Step 1: First create a file in the container

```
(py36) sh-5.0# touch TEST.txt
```

OR

```
sh-5.0# touch TEST.txt
```

Step 2: Copy the created new file from the container to the local working directory. From another termnal run,:

```
docker cp 8f780c0a9969:/root/prokseq/TEST.txt .
```

CHAPTER 3

CONDA:

Step 1: Fetch the package.:

```
conda install -p <PATH_TO_DOWNLOAD> -c <CHANNEL> prokseq
```

Example:

```
mkdir testPrseq
conda install -p /home/user/testPrseq -c snandids prokseq
```

Step 2: Once the package is obtained, run the following commands.:

```
tar -xvzf exampleFiles.tar.gz
```

Step 3: Install dependencies.:

```
tar -xvzf depend.tar.gz
```

A depend folder will be created with all the required dependencies. Most of these packages were compiled on architecture x86_64 Fedora 31. Users may have to recompile some of them.

Install the R and the R bioconductor packages.

Though the pipeline is written in Python3.6, but some packages included in the pipeline require Python2.7. Therefore, it is advised to install Python2. The program should find python2 and python (python3) in the env PATH. To make life easier, we recommend create environment (Step 4).

Step 4: Create virtual environment:

```
conda create -n yourenvname python=3.6
conda activate yourenvname
conda install pandas
pip2.7 install numpy
pip2 install qc bitsets RSeQC
pip2 install --upgrade cython bx-python pysam RSeQC numpy
```

Step 5: Once all the dependencies and R packages are installed, and the example files are untared, change the PATH in parameter file (Eg. param.input.bowtie). The PATH should point to the packages.

For example: If you are using the above-mentioned path *[/home/user/testPrseq]* from **Step 1**, then specify the path as below for all the packages in the parameter file.

```
#       Specify the path to pypy required for running afterqc
PATH PYPY /home/path/testPrseq/depend/pypy2.7-v7.2.0-linux64/bin
```

Then run the following command to test run the pipeline.:

```
python scripts/pipeline-v2.8.py -s samples.bowtie.PEsample -p param.bowtie.yaml -n 4
```

Description: The script is running with PE (paired-end) samples described in samples.bowtie.PEsample, and with the parameters defined in param.bowtie.yaml. The program is submitted with four processors.

To remove::

```
conda remove -p /home/path/testPrseq prokseq
```

CHAPTER 4

CHECK TEST RUN:

After setting up of the depend directory, one can check if the environment is all setup. The required fastq and other required files for the check run is bundled in exampleFiles.tar.gz. Therefore, this file should be untared or at least should be there in the current working directory. The check script can be run as follows.:

```
python scripts/runCheck.py
```

Note: This will require the files from exampleFiles.tar.gz.

On successful test run, an Output directory will be produced.

CHAPTER 5

DIRECTORY LAYOUT:

Default directory layout should look like below:

```
./README.md
./depend/setup.sh
./depend/README
./depend/afterqc
./depend/bowtie2
./depend/FastQC
./depend/pypy2.7-v7.2.0-linux64
./depend/readFasta
./depend/RSeQC-2.6.2
./depend/salmon-latest_linux_x86_64
./depend/samtools
./depend/samtools-1.10
./depend/subread-1.4.6-p5-Linux-i386
./depend/wigToBigWig
./scripts/prokseq.py
./scripts/runCheck.py
./scripts/GraphicalAbstractProkSeq.png
./scripts/plotScript.R
./scripts/gff3_2_gtf.sh
./scripts/gtf2bed.sh
./scripts/setup.sh
./scripts/samtools.sh
./scripts/libmod
./scripts/libmod/checkEnv.py
./scripts/libmod/errMsgFn.py
./scripts/libmod/execCmd.py
./scripts/libmod/__init__.py
./scripts/libmod/pipeFunc.py
./scripts/libmod/__pycache__
```

CHAPTER 6

EXAMPLE FILES:

Example files layout:

```
./sampleCtrl_1.R1.fq
./sampleCtrl_1.R2.fq
./sampleCtrl_2.R1.fq
./sampleCtrl_2.R2.fq
./sampleCtrl_3.R1.fq
./sampleCtrl_3.R2.fq
./sampleTreat_1.R1.fq
./sampleTreat_1.R2.fq
./sampleTreat_2.R1.fq
./sampleTreat_2.R2.fq
./sampleTreat_3.R1.fq
./sampleTreat_3.R2.fq
./samples.bowtie.PEsample
./samples.bowtie.SEsample
./samples.salmon.PEsample
./samples.salmon.SEsample
./param.bowtie.yaml
./param.salmon.yaml
./oldAnnotationGFF.bed
./oldAnnotationGFF.gtf
./orf_coding_all.fasta
./SequenceChromosome.fasta
./data/TERM2GENE.csv
./data/TERM2NAME.csv
./testFile.bgl
```

CHAPTER 7

REQUIRMENTS:

Users can run the ProkSeq program to see which depending packages are missing. By default, ProkSeq will search for required programs and print the availability of the program.

ProkSeq requires the following packages:

7.1 EXTERNAL TOOLS:

```
Package/program :       Purpose
---------------         -------
FastQC          :       Quality check
Bowtie          :       Aligning the reads
Pypy            :       For speed and memory usage we sometimes
                        uses pypy an alternative implementation
                        of python 3.6
featureCounts
from subread    :       Counting reads to genomic features such as
                        genes, exons, promoters, and genomic bins.
AfterQc         :       Automatic filtering trimming of the fastq
                        sequences.
Samtools        :       For post-processing of the SAM and BAM
                        reads alignment files.
Salmon          :       A tool for wicked-fast transcript
                        quantification from RNA-seq data.
```

The dependencies mentioned above are essential. The executable binaries are bundled in the folder "depend" in Docker and Anaconda repositories. If the user is fetching the package from github [https://github.com/snandiDS/prokseq-v2.0], then the user will get a script [setup.sh] inside the depend folder. Please run this script as:

```
sh setup.sh
```

This script will fetch the required dependencies from http://www.fallmanlab.org. The script will also ask if the user wants to compile samtools. After running the script and accepting Y for compiling the samtools, the output on the screen would be as follows:

```
-- Viewing
   flags          explain BAM flags
   tview          text alignment viewer
   view           SAM<->BAM<->CRAM conversion
   depad          convert padded BAM to unpadded BAM
```

This means the program ran successfully.

7.2 R packages:

```
**R packages**  :       **Purpose**

ggplot2         :       A system for declaratively creating graphics,
                        based on The Grammar of Graphics.

**Bioconductor Packages** :  **Purpose**

DESeq2          :       Differential gene expression analysis based
                        on the negative binomial distribution.
edgeR           :       Package for examining differential expressi-
                        on of replicated count data.
NOISeq          :       A non-parametric approach for the different-
                        ial expression analysis of RNseq-data.
limma           :       A package for the analysis of gene expressi-
                        on data arising from microarray or RNA-seq
                        technologies.
clusterProfiler :       To analyze functional profiles of genomic
                        coordinates (supported by ChIPseeker), gene
                         and gene clusters.
apeglm          :       The adaptive t prior shrinkage estimator
                        used to Shrink log2 fold changes.
RUVSeq          :       Remove Unwanted Variation from RNA-Seq Data
RColorBrewer    :       Required to create nice looking color palettes
                        especially for thematic maps, used with RUVSeq.
reshape2        :       To transform data between wide and long formats.
```

7.3 PYTHON LIBRARIES:

This program is written in python 3.6, and uses the following python libraries. os, subprocess, re, pandas, optparse, math, threading. These libraries may be installed using pip.

CHAPTER 8

RUNNING PROKSEQ:

8.1 SYNTAX:

```
Usage: prokseq.py [options] arg
Options:
  -h, --help            show this help message and exit
  -s SAMPLE_FILE_NAME, --sample=SAMPLE_FILE_NAME
                        provide the sample file
  -p PARAMETER_FILE_NAME, --param=PARAMETER_FILE_NAME
                        provide the parameter file
  -n NUMBER OF PROCESSORS, --numproc=NUMBER OF PROCESSORS
                        provide the number of processors
  -v, --version         Version of the package
```

8.2 EXAMPLE:

```
python scripts/prokseq.py -s samples.bowtie.PEsample -p param.bowtie.yaml -n 4 (For␣
↪paired end reads)

python scripts/prokseq.py -s samples.bowtie.SEsample -p param.bowtie.yaml -n 4 (For␣
↪single end reads)
```

The program will run with sample file "samples.bowtie.PE/SEsample", and parameter file "param.bowtie.yaml". The program will also utilize 4 processors.

samples.bowtie.PEsample: This is for Paired end reads

samples.bowtie.SEsample: This is for single end reads

To run the program, the dependencies mentioned above are essential. However, the executable binaries are bundled in the folder "depend". The details of the parameter and the sample files are as below. An example parameter (param.bowtie.yaml) and sample file (samples.bowtie.PEsample) are bundled together with the package in the exampleFiles.tar.gz.

CHAPTER 9

PARAMETER FILE:

There should be one parameter file. The entries of the file should be as follows.

```
###################################################################
#       File "param.yaml" - definition file in YAML format. Define
#       the paths and parameters to run the external packages. All
#       the default parameters are considered. However, if users wish
#       to modify any default parameters, please specify here.
#       NOTE: Any flag/parameter which do not require any value,
#       specify "TRUE"/"FALSE." Quotes are necessary. "TRUE" to
#       invoke the flag and "FALSE" to suppress.
###################################################################
#       Describe the Bowtie options below as:
BOWTIE:
   -I : 0
   -X : 500
   -k : 1
   -p : 1
#       In case the package salmon is to be run, uncomment the
#       following options. These are the default values.
#       These options will overwrite the BOWTIE parameters.
#
#SALMON:
#   SALMONINDEX:
#      -k : 29
#   SALMONQUANT:
#      -l : A
#      -p : 2
#      --validateMappings : "TRUE"
#
#       Define the parameters for AfterQC. Example default parameters
#       are shown as bellow. All the default parameters are taken.
#       Specify the parameter and value in case of any changes. One can
#       add the other parameters with value.
#AFTERQC:
```

(continues on next page)

(continued from previous page)

```
#   -s : 35
#   -n : 5
#   -p : 35
#   --debubble : "TRUE"
#   -a : 5
#
#       Define the Featurecounts options as bellow:
FEATURECOUNTS:
   a : oldAnnotationGFF.gtf #Define the Featurecounts input GTF file.
   o : FeatCount #Define the Featurecounts output file name.
#
#       Specify a count file name
COUNTFILE : countFile.csv
#
#       geneBody coverage.r require a bed file. Specify the name of bed file as␣
↪bellow:
geneBody_coverage:
   r : oldAnnotationGFF.bed
#
#       Specify if batch effect removal is required. FALSE if not required, else TRUE.
BATCH_EFFECT_REMOVE : "FALSE"
#
#Remove Unwanted Variation from RNA-Seq Data
#To run RUVSeq, uncomment the following two lines.
RUVSeq:
   nc : 100 #Integer - Top 100 genes as ranked by edgeR p-values. Negative control␣
↪genes to estimate the factors of unwanted variation.
#
#       Parameters for pathway analysis.
#       For pathway analysis, define the organism in three alphabets as bellow.
#       ypy = Yersinia pseudotuberculosis
#       User need to change the keg abbreviation of their genome which can be
#       found in https://www.genome.jp/kegg/catalog/org_list.html. Here ypy is
#       the Yersinia pseudotuberculosis YPIII
#       If PATHWAY analysis is not required comment out the PATHWAY section.
PATHWAY:
   cutoffPositive : 2.0  #logfold upper limit
   cutoffNegative : -2.0 #logfold lower limit
   Organism : ypy
   # For Gene Ontology of the pathway analysis, define GO term and gene name file.
   TERM2GENE : data/TERM2GENE.csv
   TERM2NAME : data/TERM2NAME.csv
#
#       Specify the root path. That means where the ProkSeq bundle is unpacked.
#       The PATH ROOT : should indicate the current working directory.
#       The location should have the following folders
#       1. depend - contains all the binaries of the external packages
#       2. scripts - contains all the modules required for ProkSeq, and pipeline-vx.x.
↪sh
#       3. data - Contains Gene ontology files for pathway analysis.
#       4. Fastq files, and other genome/transcript files.
#       If the package is stored in the folder/path /home/user/PROKSEK
PATH ROOT : /home/user/PROKSEK
#       If the above environment (depend, scripts, data) is true, the following
#       line maye uncommented.
#PATH DEFAULT : "TRUE"
#       Specify the path to samtools
```

(continues on next page)

(continued from previous page)

```
PATH SAMTOOLS : /home/user/PROKSEK/depend/samtools/bin
#       Specify the path to geneBody_coverage
PATH geneBody_coverage : /home/user/PROKSEK/depend/RSeQC-2.6.2/scripts/
#       Specify the path to FEATURECOUNTS
PATH FEATURECOUNTS : /home/user/PROKSEK/depend/subread-1.4.6-p5-Linux-i386/bin/
#       Specify the path to fastqc
PATH FASTQC : /home/user/PROKSEK/depend/FastQC
#       Specify the path to bowtie
PATH BOWTIE : /home/user/PROKSEK/depend/bowtie2/bowtie2-2.3.5.1-linux-x86_64
#       Specify the path to salmon if salmon is required
#PATH SALMON : /home/user/PROKSEK/depend/salmon-latest_linux_x86_64/bin
#       Specify the path to pypy required for running afterqc
PATH PYPY : /home/user/PROKSEK/depend/pypy2.7-v7.2.0-linux64/bin
#       Specify the path to readfasta
PATH READFASTA : /home/user/PROKSEK/depend
#
#       End of file "param.input"
#
```

In general, the entries starting with BOWTIE instructs the program to run the tool with the additional parameter. Similarly SALMON, AFTERQC, FEATURECOUNTS, etc. Entries beginning with PATH indicates the path to the executables of the external tools. If one uses the bundled packages in the depend folder, *PATH DEFAULT : TRUE* line should be uncommented. Note *PATH ROOT : path_to_the_current_working_directory* should be mentioned.

NOTE: The program will override the Bowtie options, and the package salmon will be run if both bowtie and salmon's options are provided. Therefore, if salmon is required, please comment on the Bowtie option lines, and vice versa.

CHAPTER 10

SAMPLE FILE:

Sample file is required for the program. This file should have the following format. Please don't change the format of the file. Simply replace the fastq, sam and the conditions of the sample.

```
####################################################################################
#       File "sample" - sample description file
#       Specify the names of the sample files and tag them as "treat" and "control".
####################################################################################
#       Specify the genome file, and specify the path where the
#       indexed file will be stored, and the prifex of the indexed genome.
#       Default is 'bowtie2_genome'.
GENOME SequenceChromosome.fasta bowtie2_genome/sequenceChr
#       Specify the fastq files
#       Specify the output name of the sam files.
#       Followed by the tag/class/condition of the sample (treated or control)
#       List all the fastq files as below.
FASTQ sampleTreat_1.R1.fq sampleTreat_1.R2.fq sampleTreat_1.sam treat
FASTQ sampleTreat_2.R1.fq sampleTreat_2.R2.fq sampleTreat_2.sam treat
FASTQ sampleTreat_3.R1.fq sampleTreat_3.R2.fq sampleTreat_3.sam treat
FASTQ sampleCtrl_1.R1.fq sampleCtrl_1.R2.fq sampleCtrl_1.sam control
FASTQ sampleCtrl_2.R1.fq sampleCtrl_2.R2.fq sampleCtrl_2.sam control
FASTQ sampleCtrl_3.R1.fq sampleCtrl_3.R2.fq sampleCtrl_3.sam control
#
#       End of file "sample"
#
```

In general, this file starts with GENOME entry. As in the example, the genome file to be used in the analysis is SequenceChromosome.fasta. The next argument indicates what would be the prefix of the indexed genome, and where to store.

The following lines are FASTQ. The first argument in the entry is forward fastq file, and second is the reverse fastq file. However, if one has only single-end reads, only one entry may be there. The subsequent argument is the SAM file name. This argument specifies the output name of the sam files after running the bowtie. In this example, sampleTreat_1.sam, sampleTreat_2.sam, etc. are the sam files for sampleTreat_1.R1/R2.fq, sampleTreat_2.R1/R2.fq, etc. The last argument is the condition/class of the sample file (example: “treat” and “control”).

In case of SALMON:

```
##################################################################################
#       File "sample" - sample description file
#       Specify the names of the sample files and tag them as "treat" and "control".
##################################################################################
#       Specify the genome file and specify the path where the
#       indexed file will be stored, and the prifex of the indexed genome.
#       Default is 'transcripts_index'.
GENOME orf_coding_all.fasta transcripts_index
#       Specify the fastq files
#       Specify the output name of the quant files.
#       Followed by the tag/class/condition of the sample (treated or control)
#       List all the fastq files as bellow.
FASTQ sampleTreat_1.R1.fq sampleTreat_1.R2.fq sal_quant1 treat
FASTQ sampleTreat_2.R1.fq sampleTreat_2.R2.fq sal_quant2 treat
FASTQ sampleTreat_3.R1.fq sampleTreat_3.R2.fq sal_quant3 treat
FASTQ sampleCtrl_1.R1.fq sampleCtrl_1.R2.fq sal_quant4 control
FASTQ sampleCtrl_2.R1.fq sampleCtrl_2.R2.fq sal_quant5 control
FASTQ sampleCtrl_3.R1.fq sampleCtrl_3.R2.fq sal_quant6 control
#
#       End of file "sample"
#
```

In general, this file starts with GENOME entry. As in the example, the transcript file to be used in the analysis is transcripts.fasta. The next argument indicates what would be the prefix of the indexed file, and where to store.

The following lines are FASTQ. The first argument in the entry is forward fastq file and second is the reverse fastq file. However, if one has only single-end reads, only one entry may be there. The subsequent argument is the directory where the alignment file has to be stored. The last argument is the condition/class of the sample file (example: “treat” and “control”).

CHAPTER 11

DATA FILES:

```
1. Samples files in fastq format.
2. Pathway analysis (Optional):
        1. TERN2NAME.csv
           Gene Ontology to terms mapping csv file (Eg: GO:0000001,mitochondrion␣
↪inheritance).
           This is the GO term classification which is common for all organisms.
        2. TERM2GO.csv
           Gene Ontology to gene mapping csv file (Eg: GO:0003688,YPK_0001). This is␣
↪Genome
           specific Gene ontology file. TERM2GENE.csv is a comma delimited 2 column␣
↪file.
           First column is the GO term and second column is the gene name. User can␣
↪download
           the GO file or GFF annotation file from `_[Genome2D webserver]
           <http://genome2d.molgenrug.nl/g2d_core_select_genbank.php)>`_,

3. For Bowtie implementation:
      Genome file in fasta format
   For Salmon implementation:
      Transcript file in fasta format
4. GTF file
- Bacterial GTF files can be downloaded from the following link `_[GTF files]
  <https://bacteria.ensembl.org/info/data/ftp/index.html>`_,
5. Bed file
-Bed file can be produce by using the gtf2bed.sh script found in the ProkSeq folder
```

All these files should be declared in SAMPLE FILE and PARAMETER file.

We have deposited 15 bacterial genome sequence and annotation files in the following link
Acinetobacter_baumannii_ATCC,
Neisseria_gonorrhoeae_FA_1090,
Campylobacter_jejuni_CJ677CC541,

Escherichia_coli_O157,
Klebsiella_pneumoniae_ATCC,
Streptococcus_pyogenes_ATCC,
Yersinia_enterocolitica_LC20,
Yersinia_pestis_Antiqua,
Vibrio_cholerae_MJ_ASM2258v1,
Staphylococcus_aureus_ASM59796v1,
Salmonella_enterica_subsp_enterica_serovar_Typhimurium,
Pseudomonas_aeruginosa_PAO1,
Mycobacterium_tuberculosis,
Listeria_monocytogenes_07PF0776,
Helicobacter_pylori

To use these files, user need to follow the following.

For example using *Mycobacterium tuberculosis*:

```
> wget http://www.fallmanlab.org/wp-content/uploads/2020/09/Mycobacterium_
↪tuberculosis.zip
> unzip Mycobacterium_tuberculosis.zip
> cd Mycobacterium_tuberculosis
> sh /home/path/testPrseq/scripts/gff3_2_gtf.sh -f Mycobacterium_tuberculosis_variant_
↪bovis_BCG_str_ATCC_35743_63839_ASM19407v3_genomic.gff > myco_tub.gtf
> sh /home/path/testPrseq/scripts/gtf2bed.sh -f myco_tub.gtf > myco_tube.bed
```

Once generated the gtf and the bed files, user need to include these files in the parameter file.

```
#       Define the Featurecounts options as bellow:
FEATURECOUNTS:
   a : myco_tub.gtf #Define the Featurecounts input GTF file.
#       geneBody coverage.r require a bed file. Specify the name of bed file as␣
↪bellow:
geneBody_coverage:
   r : myco_tube.bed
```

*Also the **TERM2GENE.txt* should be copied to the **data** directory.*

CHAPTER 12

OUTPUT:

ProkSeq produces several folder with analysis results as a Output folder.

The structure of the Output directory looks like

```
QC_preFilter
QC_afterFilter
alignmentFile
countAndExpression
DiffExpResults
PathwayEnrichment
genomeBrowseFile
Plots
```

12.1 1. QC_preFilter:

- fastQC out put html file

12.2 2. QC_afterFilter:

Remove bad quality read and filter good quality reads. It contains three subfolder

- **FilteredReads**: good quality filtered fastq files
- **QCfastQ_filtered** : quality checking html file of filtered reads
- **RemovedReads**: Bad quality reads that is removed for analysis

12.3 3. alighmentFile:

- **.sam**: Sequence Alignment file generated from bowtie2/salmon

- **sam.alignSummary**: Alignment summary information/statistics of the alignment of individual sample to the reference genome.

12.4 4. countAndExpression:

Depending on the file names provided in the parameter file for the 'Featurecounts output file' and 'COUNTFILE'.

- **Count.csv** : The file contains total count according to genomic features
- **countFile_TPM_CPM.csv** : The file contains total count according to genomic features as well as Count per miillion(CPM), and Transcript per million (TPM)
- **countFile.NucleotideAvgCount.csv** : THis file contains Average nucleotide expression per gene as well as other total count, CPM and TPM.

12.5 5. DiffExpResults:

- **DESeq2_results.txt**: Differential expression results from DESeq2
- **DESeq2lfcShrink_results**: Differential expression results from DESeq2 with Log2 Fold Shrinkage
- **edgeR_results.txt**: Differential expression results from edgeR
- **afterNoiseq.txt**: Differential expression results from NOISeq
- **RUV_DESeq2_results.txt**: Differential expression results with RUV normalization by using RUVSeq and DE-Seq2

12.6 6. PathwayEnrichment

- **GOpathways.txt** : This file contains the name of the enrich GO term as well as enrichment score.
- **GOenricher.txt** : This file contains the name of the differential genes within the enriched GO terms.
- **KEGGpathway.txt** : This file contains the name of the enrich KEGG pathways as well as enrichment score.
- **KEGGenricher.txt** : This file contains the name of the differential genes within the enriched KEGG pathways

12.7 7. genomeBrowserFile:

- **bam**: Folder contain Binary alignment file (BAM) as well as sorted and indexed BAM. Users are advised to use sorted.bam file for raw aligned file visulazation by genomic browser IGV
- **.wig**: Single nucleotide visualization wiggle file for visualization or other purpose
- **.bw** : Single nucleotide visualization Big wiggle file which is memory efficient for visualization
- **normalized.wig**: Single nucleotide visualization normalized Big wiggle file if user wants to visually compare RNA-seq data of different library depth.

** All the .txt file and .csv file can be opened via XL.** For details please follow the link **'[How to open .txt and .csv file in XL] https://knowledgebase.constantcontact.com/articles/KnowledgeBase/6269-convert-a-text-file-to-an-excel-file?lang=en_US'_**

12.8 8. Plots:

Contains all the plots generated by ProkSeq during analysis in pdf, png and tiff format.

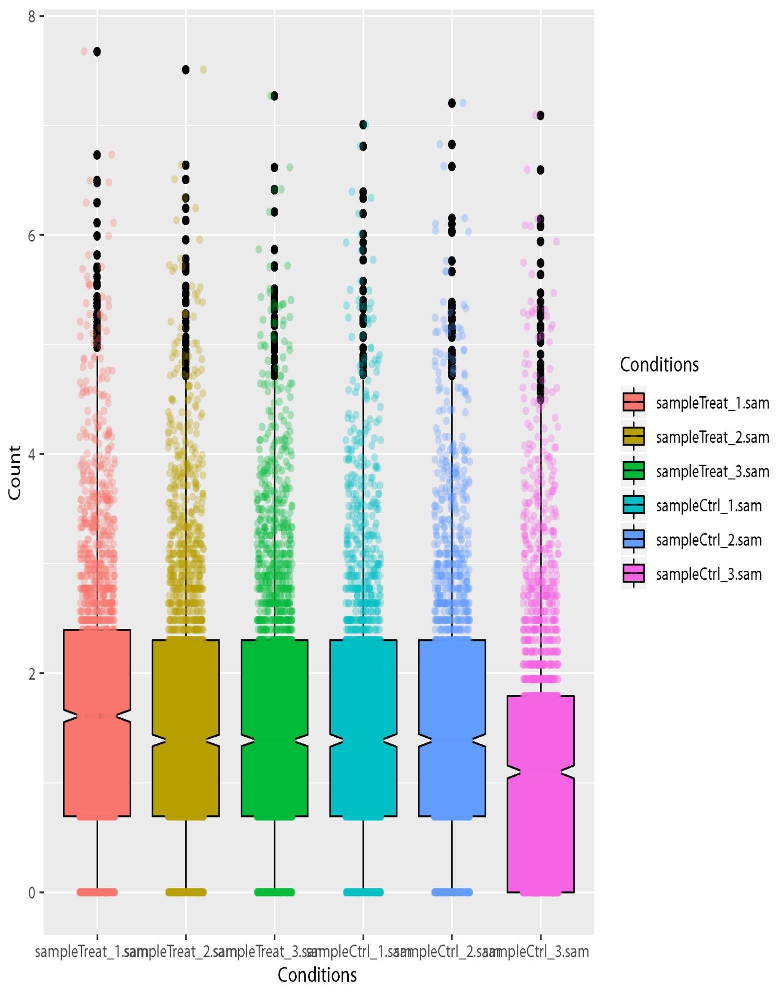
CHAPTER 13

RESULT.html

ProkSeq generates a result.html file that contains all the figures and a brief description about the analysis it done.

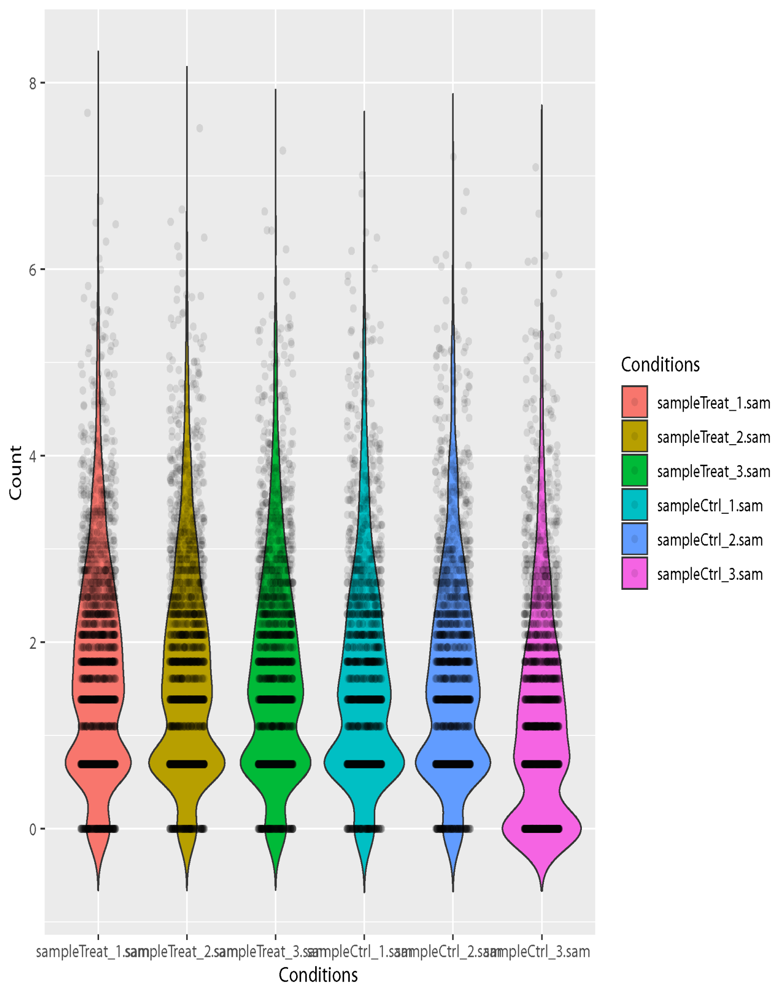
The result.html file would look like the following if user runs the sample data

Results of the anlysis The result of the analysis is below: *Sample correlation plots* Sample correlation plots: Box plot showing the reads distribution on the CDS. Y axis indicates the number of reads in a annotated feature



Sample violin plots:

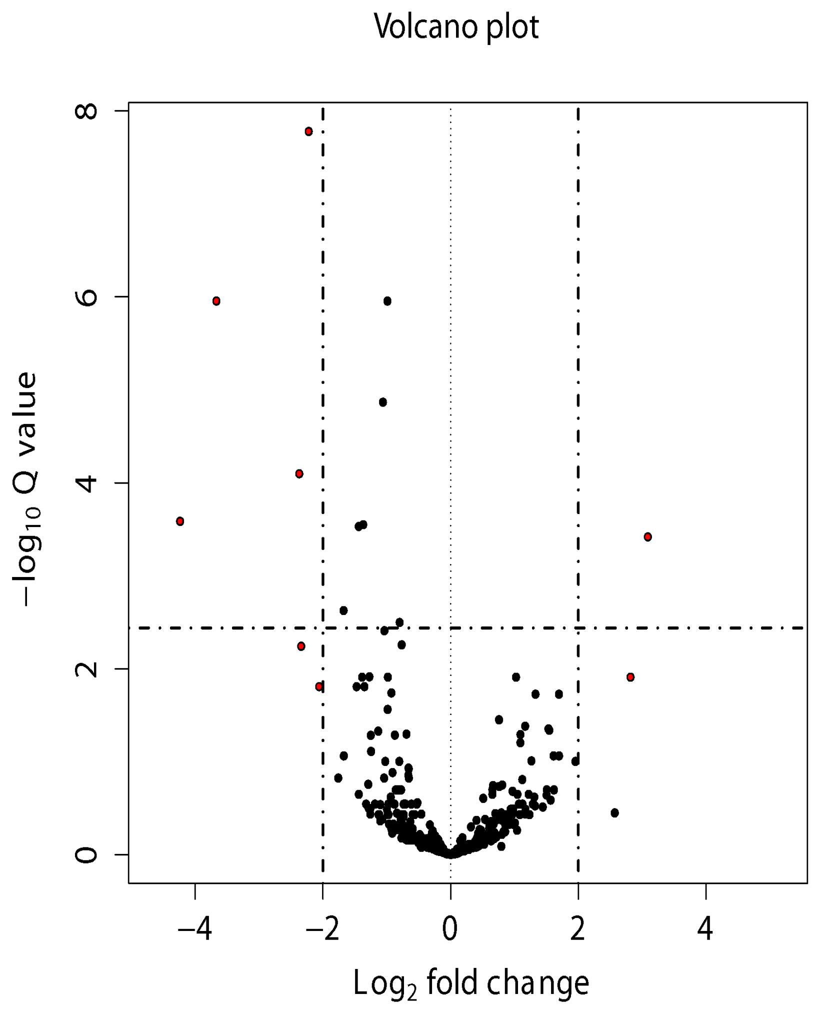
Showing the distribution shape of the reads within CDS. Wider sections of the violin plot represent a higher probability of the reads is distribute in the region.



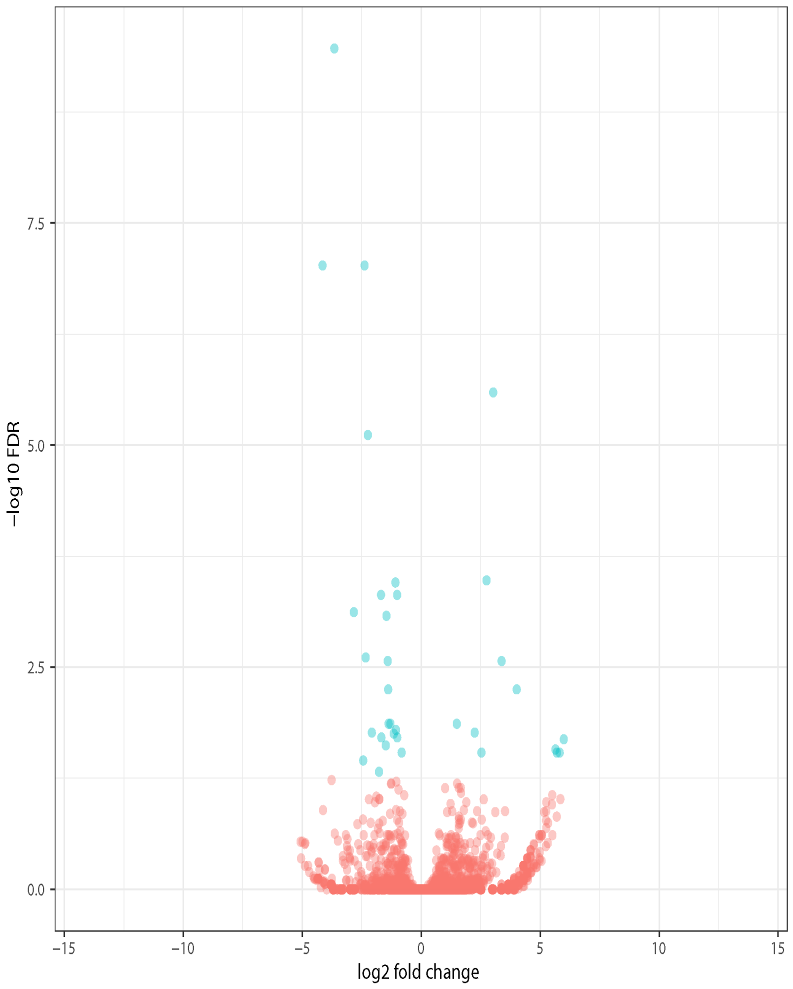
Sample volcano plots

Volcano plot shows statistical significance (P value) in Y axis and magnitude of differential expression in fold change.

DESeq2 volcano plot:

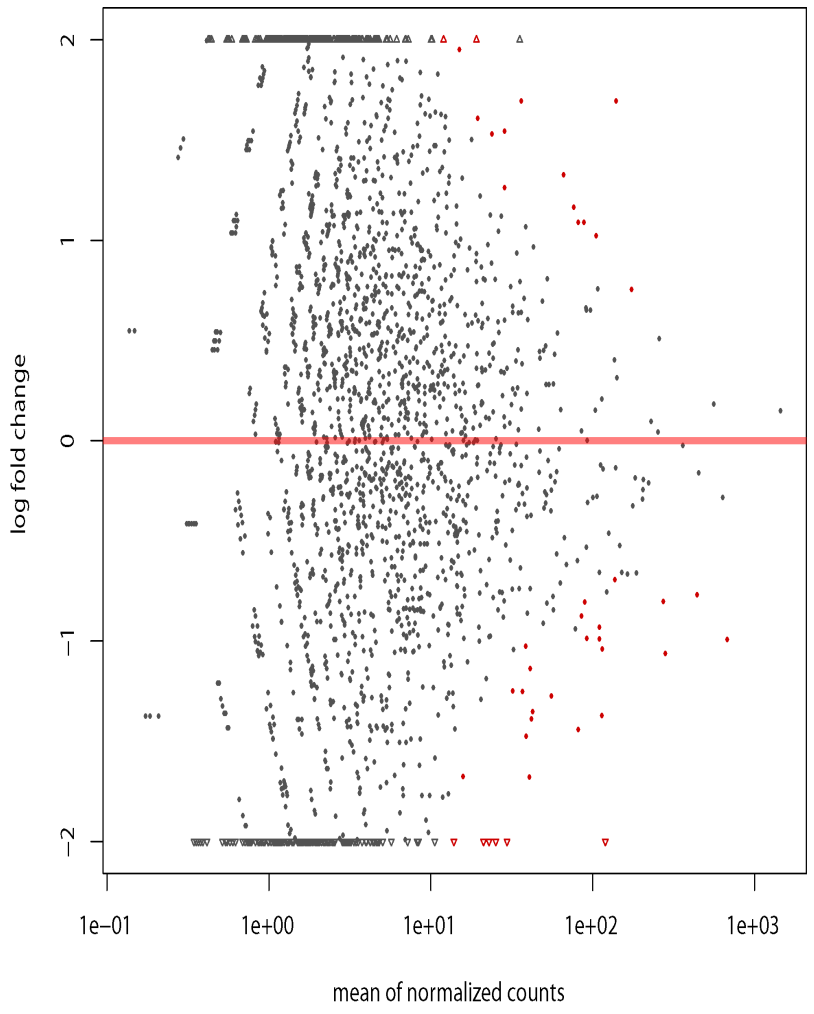


Edger volcano plot:

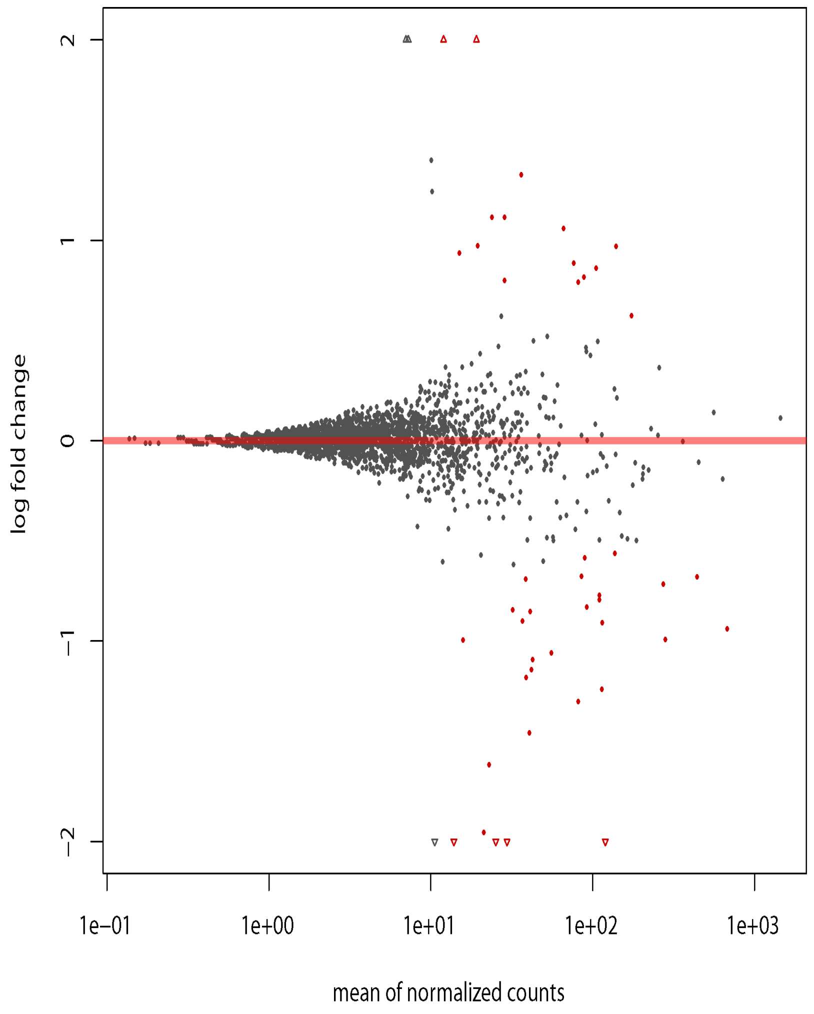


MA plot

MA plot: Showing log-fold change (M-values, i.e. the log of the ratio of level counts for each CDS between treatment and control) against the log-average (A-values, i.e. the average level counts for each CDS across treatment and control)

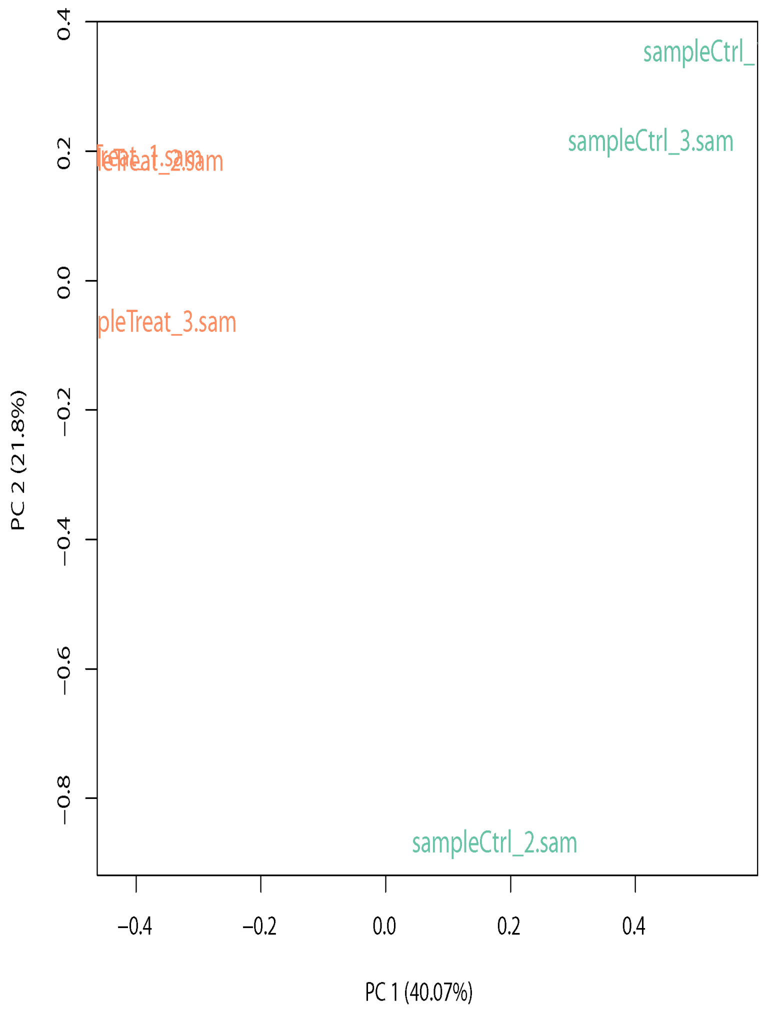


MA LFC plot: For Shrinking the log2 fold changes by using the information from all genes to generate more accurate estimates when the information for a gene is less as for example low counts, high dispersion values.



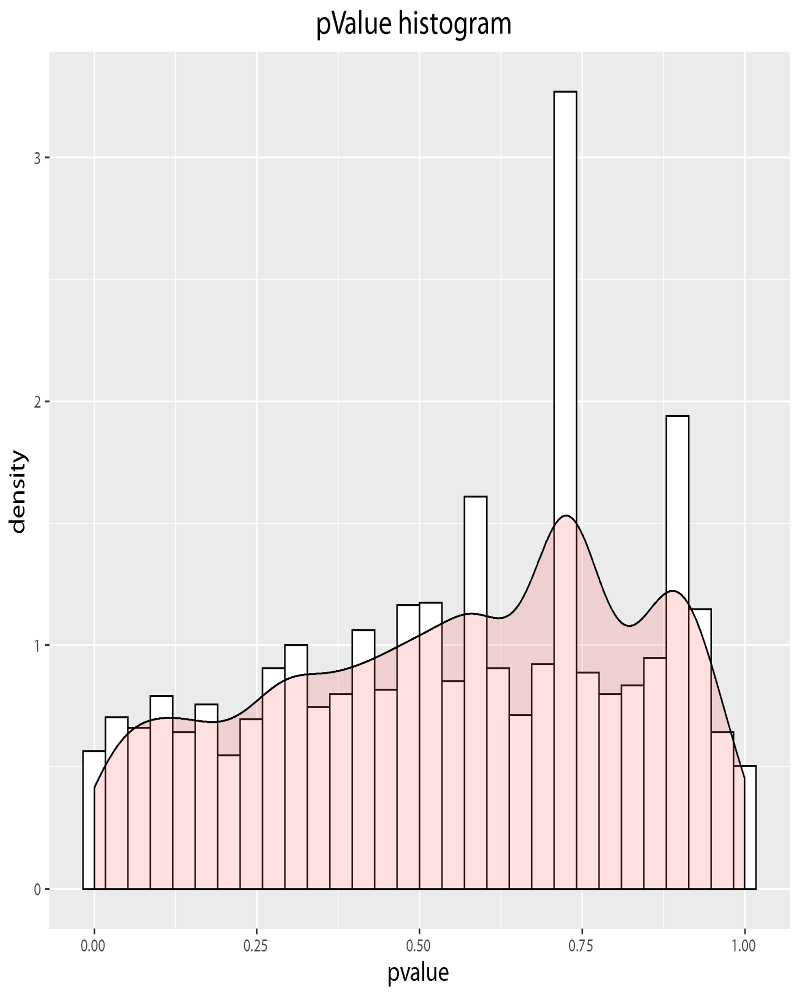
PCA plot

PCA plot: Principle component analysis (PCA) shows the variance between treatment and control and within biological replicates.



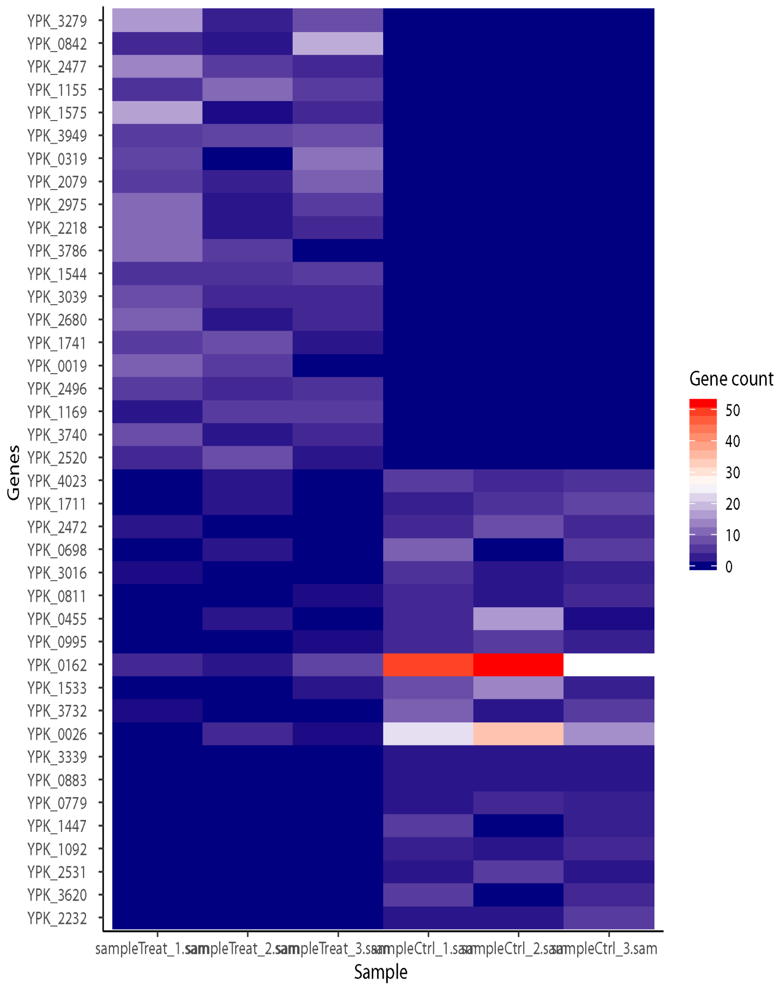
pValue-histogram plot

pValue-histogram plot: Showing the P-value distribution of the differential expressed genes. The enrichment of low p-values indicates differentially expressed genes, while those not differentially expressed are spread uniformly over the range from zero to one.



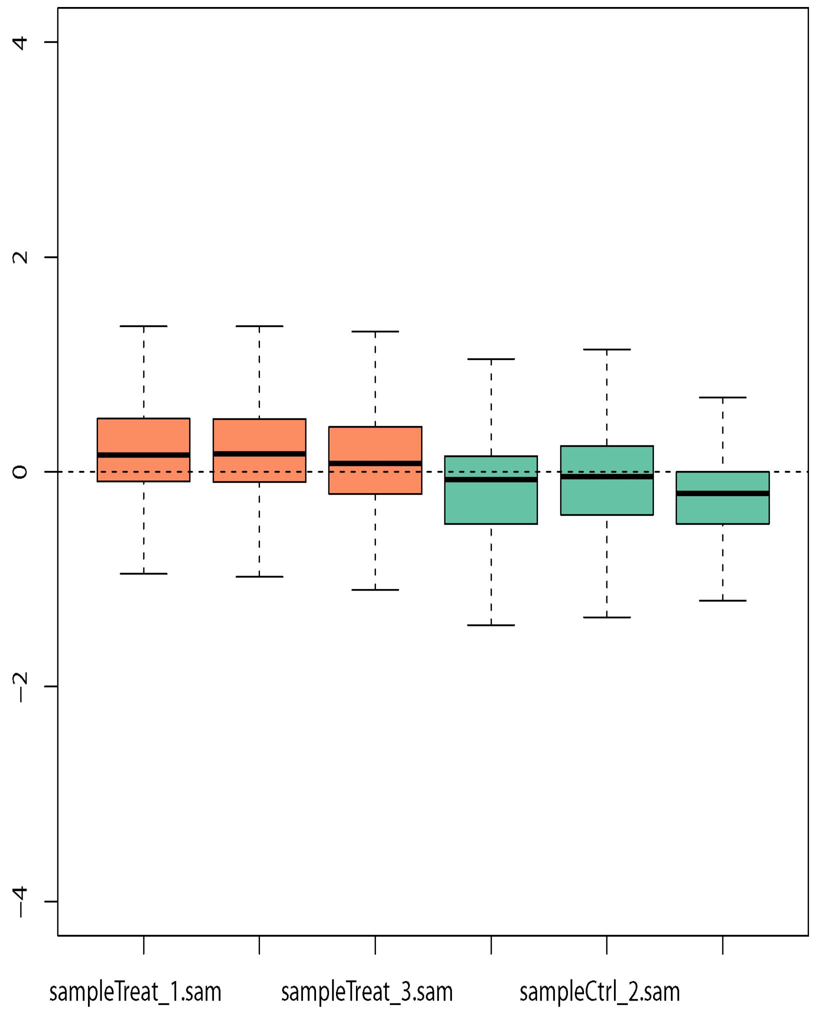
Gene count heat map

Gene count of the top 40 genes arranged based on log fold change across samples.

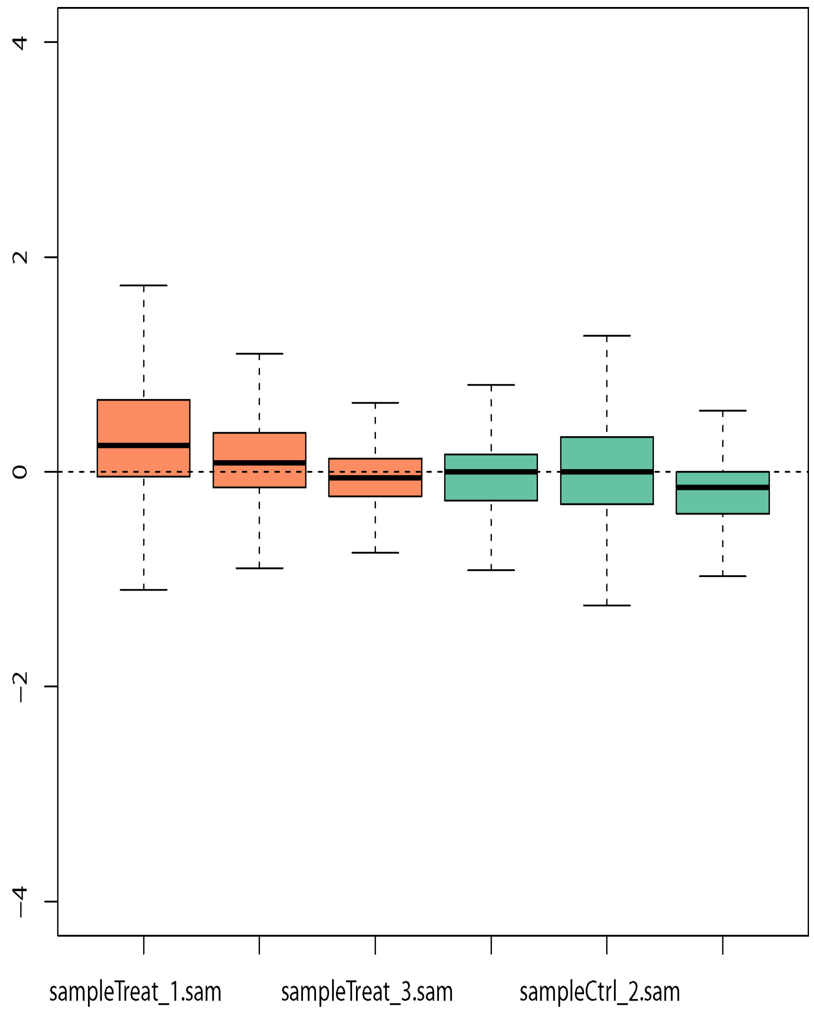


Normalization by removing unwanted variation analysis

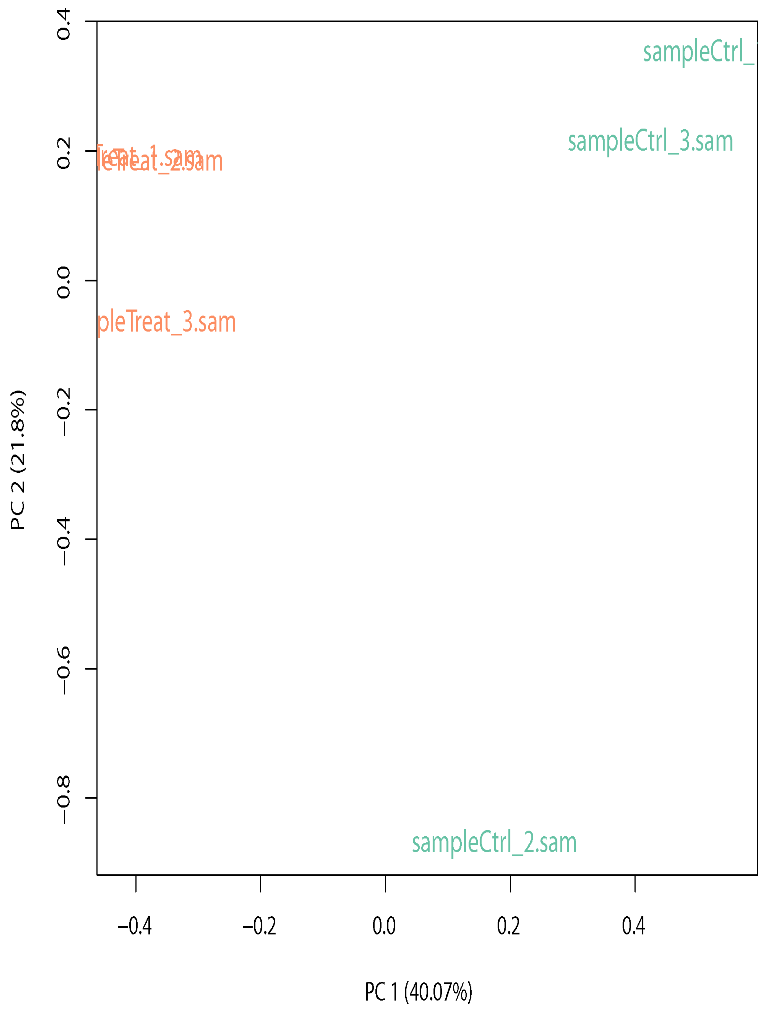
Normalization of unwanted variation using negative control genes that are not expected to be influenced by the biological covariates of interest. Boxplots of relative log expression (RLE = log-ratio of read count to median read count across sample) before removing unwanted variation:



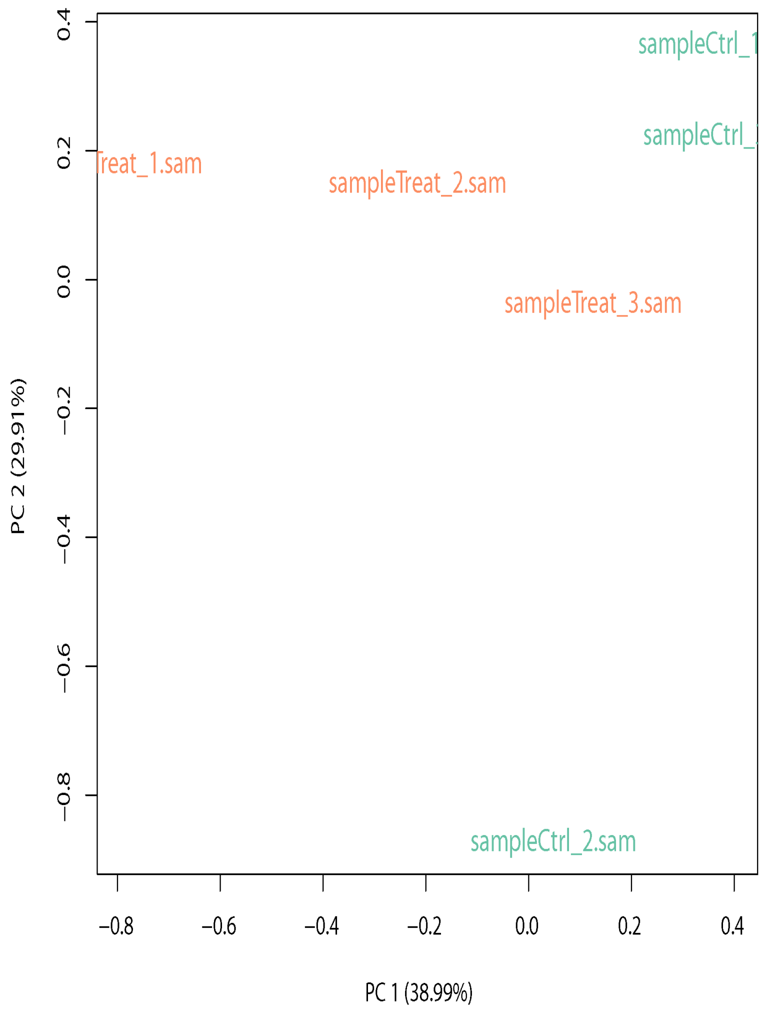
Boxplots of relative log expression (RLE = log-ratio of read count to median read count across sample) after removing unwanted variation:



Principal components plots before removing unwanted variation:

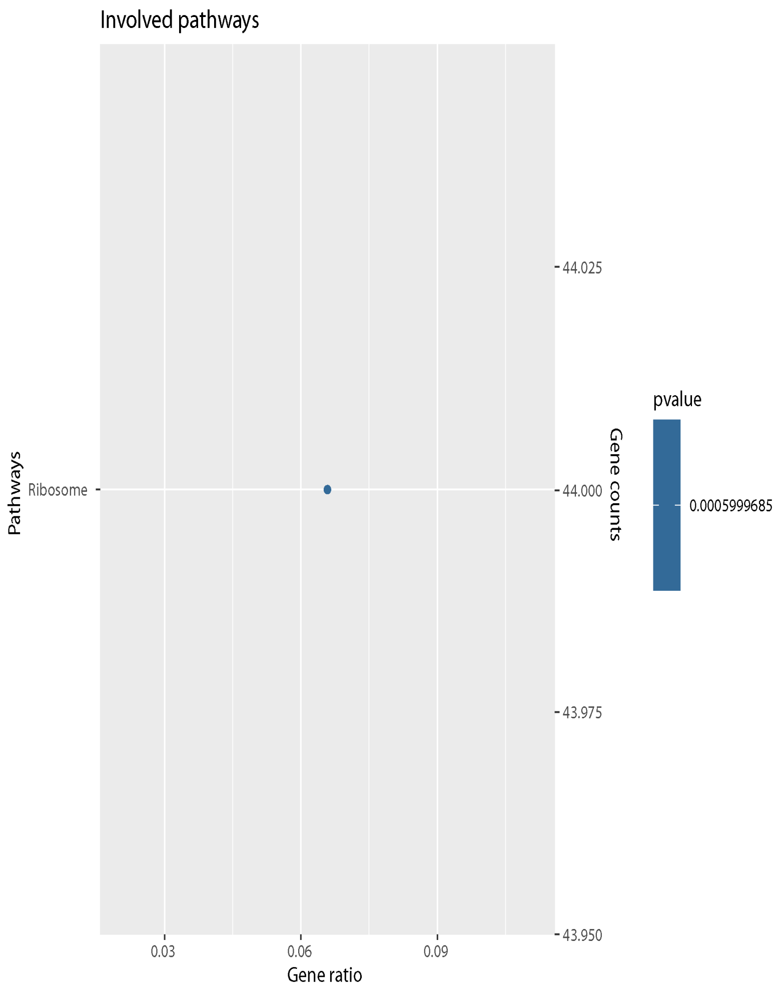


Principal components plots after removing unwanted variation:

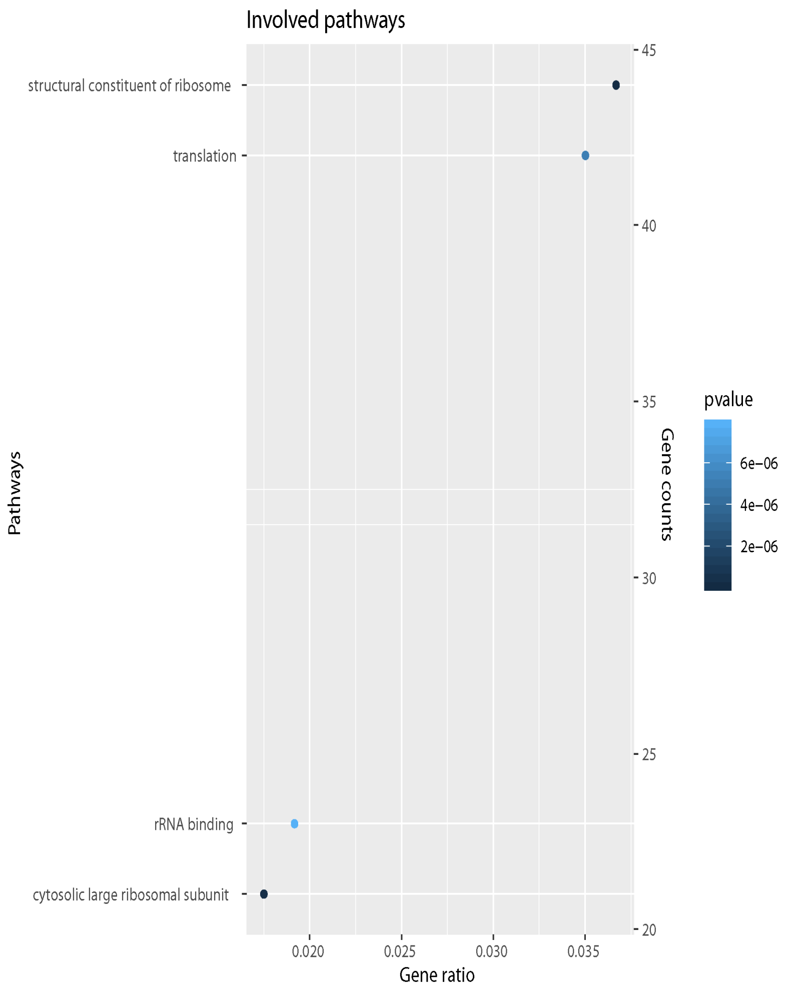


PATHWAY

KEGG


Involved pathways
Pathways
Ribosome
Gene counts
44.025
44.000
43.975
43.950
0.03
0.06
0.09
Gene ratio
pvalue
0.0005999685

GO


Involved pathways
structural constituent of ribosome
translation
rRNA binding
cytosolic large ribosomal subunit
Pathways
0.020
0.025
0.030
0.035
Gene ratio
45
40
35
30
25
20
Gene counts
pvalue
6e−06
4e−06
2e−06

CHAPTER 14

OTHER SUPPORTING PROGRAMS:

1. **gtf2bed.sh**:

   This script transform the GTF file to a bed file. Syntax:

   sh gtf2bed.sh -f gtfFile_name > bedFile_name

   Options:

-h	help message
-f	GTF file name

   Example: sh gtf2bed.sh -f annotation.gtf > annotation.bed

2. **gff3_2_gtf.sh**:

   This script transform the GFF3 file to a GTF file. Syntax:

   sh gff3_2_gtf.sh -f gffFile_name > gtfFile_name

   Options:

-h	help message
-f	GFF file name

   Example: sh gff3_2_gtf.sh -f annotation.gff > annotation.gtf

3. **plotScript.R**:

   This script will run DESeq2 and create plots from the DESeq2 results. SYNTAX:

   Rscript scripts/DESeq2Plot.R [SE/PE] sampleFile

   Example:

   Rscript scripts/DESeq2Plot.R SE samples.bowtie.SEsample The above example will run the script with the arguments SE (single-end reads), and the samples.bowtie.SEsample. The sample names will be picked up from the samples.bowtie.SEsample.

This script should be executed where the following files are available. 1. DESeqCount-Data.csv 2. DESeqCountCondition.csv 3. sampleFile (example samples.bowtie.SEsample) 4. edgeR_results.txt 5. countFile.csv 6. sampleCountNames.txt

Output: The script would create the following plots: 1. DESeqMAplot.pdf 2. DESeqMA-LFCplot.pdf 3. DESeq-Volcano.pdf 4. pValue-histogram.pdf 5. pca.pdf 6. correlation.pdf 7. correlation_violin.pdf 8. edgeRheatMap.pdf

MANAGING FIGURE's DIMENSION AND RESOLUTION: To manage the figure's properties, edit the pdf("fileName.pdf") lines with the following instructions.

1. Specify file name to save the plot [jpeg(), png(), svg() or pdf()]. Specify additional argument indicating the width, height, and the resolution of the image.
2. Create the plot with graphics command
3. Close the file with the call dev.off()

EXAMPLE: [PNG, jpeg, tiff]:

png(file="myplot.png",width=400,height=350,res=1200) plot(x,y,main="EXAMPLE PLOT") dev.off()

Vector-based format [PDF, SVG]: pdf(file="myplot.pdf",width=400,height=350,res=1200) plot(x,y,main="EXAMPLE PLOT") dev.off()

CHAPTER 15

INSTALLATION INSTRUCTION FOR THE DEPENDENCIES:

If one or any of the above dependencies are missing user can install it by following the instructions below.

15.1 Python3:

```
#Ubuntu
Ubuntu 17.10, Ubuntu 18.04 (and above) come with Python 3.6 by default.
Ubuntu 16.10 and 17.04 do not come with Python 3.6 by default, but it is in the␣
↪Universe repository.
You should be able to install it with the following commands:

- sudo apt-get update
- sudo apt-get install python3.6

For Ubuntu 14.04 or 16.04, Python 3.6 is not in the Universe repository, and user do␣
↪not need to get
it from a Personal Package Archive (PPA). For example, to install Python from the␣
↪"deadsnakes" PPA,
do the following:
- sudo apt-get update
- sudo apt-get install python3.6

#CentOS
User should first update the system with the yum package manager:
- sudo yum update
- sudo yum install yum-utils
Then install the CentOS IUS package
- sudo yum install https://centos7.iuscommunity.org/ius-release.rpm
Then install Python and Pip:
- sudo yum install python36u
- sudo yum install python36u-pip
```

15.2 Installation of R:

```
The pipeline is tested on R version 3.6.0.
#Installing R on Ubuntu 19.04/18.04/16.04
Before installing R, user need to update the system package index and upgrade all ␣
↪installed packages
using the following two commands:
-sudo apt update
-sudo apt -y upgrade
After that, run the following in the command line to install base R.
-sudo apt -y install r-base
# Install R on CentOS 7
R packages are available in the EPEL repositories. It can be installed by typing:
-sudo yum install epel-release

Once the repository is added, install R by typing:
-sudo yum install R
```

15.3 Installation of R Bioconductor packages:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
        install.packages("BiocManager")

edgeR:
BiocManager::install("edgeR")
DESeq2:
BiocManager::install("DESeq2")
NOISeq:
BiocManager::install("NOISeq")
limma:
BiocManager::install("limma")
clusterProfiler:
BiocManager::install("clusterProfiler")
apeglm:
BiocManager::install("apeglm")
RUVSeq:
BiocManager::install("RUVSeq")
```

15.4 Samtools:

```
#Ubuntu 18.04 or higher
Install samtools by entering the following commands in the terminal:
-sudo apt update
-sudo apt install samtools
For the other version of Ubuntu or centose use the samtools.sh script in the package␣
↪folder. User can
go to the PorkSeq folder and open a terminal and write sh samtools.sh. The program␣
↪will install samtools
in the samtools directory.
```

15.5 EXTERNAL TOOLS:

```
This program uses the following tools.
1. FastQC : This package runs the quality check
2. Bowtie : Needed for aligning the reads
3. Pypy : For speed and memory usage we sometime uses pypy an alternative␣
↪implementation of python 3.6
4. featureCounts from subread: a software program developed for counting reads to␣
↪genomic features such
as genes, exons, promoters and genomic bins.
5. AfterQc: Tools for automatic filtering trimming of the fastq sequences.
```

To run the program the above mentions dependencies are essential. However, the executable binaries are bundled in the folder depend. For ubuntu 18.04 or higher version use sudo apt-get update | apt-get install python3-pandas to install pandas User can download the depend folder along with all the dependencies from the following link '[Depend folder] https://umeauniversity-my.sharepoint.com/:u:/g/personal/aakk0004_ad_umu_se/EZ6UF28lCcJGiuPOWQ8oVr0BtQAK1caGUEdVHuP29_I01g?e=o1K0mh '_ OR, follow the following:

```
1. Create a folder named depend
2. cd depend
3. cp ../scripts/setup.sh .
4. sh setup.sh
```

The depend folder will be populated.

CHAPTER 16

How to Cite ProkSeq:

A K M Firoj Mahmud, Nicolas Delhomme, Soumyadeep Nandi, Maria Fällman, ProkSeq for complete analysis of RNA-Seq data from prokaryotes, Bioinformatics, 2020;, btaa1063, https://doi.org/10.1093/bioinformatics/btaa1063

Link of the article: https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/btaa1063/6050703

CHAPTER 17

FREEQUENTLY ASKED QUESTIONS (FAQ):

17.1 How can I get GTF, BED and GO annotation file of my strain?

ProkSeq provides a file that list all the bacterial GO annotation file downloadable link (Bacterial_annotation_Download_link.txt). You can download GTF and GO annotation file from the link. For BED file use the script gtf2bed.sh (you can find it in the script folder) to convert downloaded gtf file to bed. ProkSeq require two file for GO enrichment analysis 1) TERN2NAME.csv: This is the GO term classification which is common for all organisms. Can be found in ProkSeq/data folder 2) TERM2GO.csv: This is Genome specific Gene ontology file. Use the script GOannotation.py to convert the downloaded GO annotation file to required format. You can also follow other steps to get GO annotation file. First check if your strain is among the 15 bacterial species, mentioned in the readthedoc above. If so, you can download GO annotation as well as other annotation file require for ProkSeq. Secondly, you can download the GO annotation file from this link http://genome2d.molgenrug.nl/g2d_core_select_genbank.php. After that, you need to convert the format according to ProkSeq requirement (You can use XL to open the file and save as .csv format.

17.2 How the GO and KEG enrichment are done in ProkSeq?

ProkSeq uses clusterProfiler for GO and KEGG enrichment. However, ProkSeq uses the option for GO enrichment of non-model organisms in clusterProfiler. A GO annotation file for the organism can be downloaded and converted to required format by "GOannotation.sh", an added feature in ProkSeq. For KEGG pathway enrichment analysis, ProkSeq uses "enrichKEGG" option in clusterProfiler that allows to search enrich pathways in the KEGG database.

17.3 I want to do pathway analysis with less strict log2fold cut-off?

You can use change the log2fold cut off in to the parameter file. Open the parameter file and change the value accordingly PATHWAY:

cutoffPositive : 2.0 #logfold upper limit cutoffNegative : -2.0 #logfold lower limit

17.4 Should we use normalized counts for DESeq2?

- DESeq2 has assumptions, based on the property of raw unnormalised counts. So use of any normalized count like RPKM, CPM, TPM or any other normalized values is not advisable.

17.5 When can I use RUVseq Normalization?

- Risso, et al., 2014 showed that RUVSeq effectively reduces library preparation effects without weakening the sample versus control effect by using in silico empirical control genes. We also suggest to use RUVSeq when user suspect an unwanted variation among the library due to library depth, preparation or comparing to condition which are not very much similar like stress response, or growth phase variation.

17.6 Does ProkSeq use normalized value for Differential expression analysis?

-No, ProkSeq does not use any normalised count for DE analysis. Subsequent DE analysis in ProkSeq is done on the original counts which is accessible through the counts method in RUVSeq. Meanwhile RUVSeq + DESeq2 is valid because it does use the raw counts but includes a noise factor in the mode.

17.7 When and how to use average nucleotide coverage value?

Normalize base count method is a variant of the total count approach that use for normalizing gene-specific read alignments. In case of extreme conditions where data are presumed to have larger variation between two groups, statistical significance of differential expression of a gene can be further evaluated by using average normalized base count data as shown by Creecy et al., 2015 (https://pubmed.ncbi.nlm.nih.gov/25483350/). Average base counts of individual transcriptional features make all samples directly comparable when two data sets have higher variation because of the experimental setup like logarithmic phase versus stationary phase.

17.8 How can I convert the csv/txt file to XL file?

-ProkSeq provides an R script (Convert2XL.R) to convert all csv/txt file into XL. See the readthedoc for in details.

17.9 When can I use NOIseq?

-If under some circumstances, you do not have any biological replicate, then you can use NOISeq. For details read the paper about NOISeq https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4666377/

17.10 I do not have replicate, Can I use DESeq2 or edgeR?

-NO, we do not recommend DESeq2 or edgeR for differential expression analysis if you do not have any biological replicate.

17.11 How can I get my pictures in higher resolution, different format and size?

-ProkSeq provides an R script (plotScript.R) to do so. You can change size, resolution, type (pdf,png, tiff, svg etc). For details see OUTPUT/plot in the readthedoc.